Debugging Process
CS398
Warmup

Many students from several countries A/B/C take a test with N questions. We assume they learn as they go.

Task: Can you separate out average prior knowledge from ability to learn?

Can you construct an experiment to validate/invalidate models?
Warmup

Task: Can you separate out average prior knowledge from ability to learn?

Can you construct an experiment to validate/invalidate models?
Final Projects

http://bit.ly/cs398-m1
Open Research Problems
Open Research Problems

Feedback on the process of how someone solves a problem
Open Research Problems
Unachieved goal: understand a student based on their process
Story 1: Process matters!

(We are providing feedback on final solutions)
2. THE FORM OF THE GENERATIVE THEORY

As mentioned above, the generation of a bug has two parts: generation of an incomplete procedure and generation of a repair to any impasse that that procedure may encounter. Repair Theory defines the set of incomplete procedures by applying a set of deletion principles to a formal representation of the correct procedure. The set of repairs is defined by a set of repair heuristics and a set of critics in the following manner. When an incomplete procedure is applied to a problem and reaches an impasse, a set of repairs is performed by a generate and test problem solver.
Chapter 3: Helping Student Teachers

Presented at SIGCSE 2019

Lisa Yan

Annie Hu
Students should learn the *process* of how to solve programming problems.
But we *aren’t* providing feedback on *process*.

(We are providing feedback on final solutions)
Pensieve Tool
Students scoring in 99th percentile on midterm exam

Students scoring in ≤3rd percentile on midterm exam

Error Fraction of commits

- Other/Off-track (1, 8, 15, 16)
- Stage 1: single row (2, 3, 4)
- Stage 2: nested loop (5, 7)
- Stage 3: adjusting nested offset (6, 9, 10, 11)
- Stage 4: adding final details (12, 13, 14)
- Error
Students work faster and learn more

- **Pensieve Intervention**
  - Baseline: 7.0 hours
  - Exp Term: 6.3 hours

- **Hours Spent Coding (minus baseline)**
  - Homework: 1, 2, 3
  - 0.0, -0.5, -1.0

- **Midterm Ability**
  - 0.0, 0.1, 0.2
  - 0, 5, 10

- **Probability Mass**
  - Baseline
  - Exp Term

Table:

<table>
<thead>
<tr>
<th>Homework</th>
<th>Hours Spent Coding (minus baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.0 hours</td>
</tr>
<tr>
<td>2</td>
<td>6.3 hours</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Story 2: It is not so easy for computers to do the same task
It was my original motivation for research
Story 3: Process has statistical signal
Can we provide feedback by dynamic analysis?

- Starter code
- First attempt
- Final solution
Each node is a unique partial solution.

Pink dots are students.

Each edge is what a teacher suggested.

Solution
Challenge: can you use process to predict next step?
## Evaluation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$P_A$ Accuracy</th>
<th>$P_B$ Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>8.2%</td>
<td>4.3%</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>49.5%</td>
<td>33.6%</td>
</tr>
<tr>
<td>Min Time</td>
<td>67.2%</td>
<td>42.2%</td>
</tr>
<tr>
<td>Rivers Policy†</td>
<td>72.9%</td>
<td>78.2%</td>
</tr>
<tr>
<td>Expected Success</td>
<td>77.9%</td>
<td>56.2%</td>
</tr>
<tr>
<td>MDP†</td>
<td>80.5%</td>
<td>47.6%</td>
</tr>
<tr>
<td>Most Common Next</td>
<td>81.1%</td>
<td>49.0%</td>
</tr>
<tr>
<td>Static Analysis†</td>
<td>86.3%</td>
<td>-</td>
</tr>
<tr>
<td>Most Popular Path</td>
<td>88.3%</td>
<td>52.8%</td>
</tr>
<tr>
<td>Ability Model</td>
<td>88.4%</td>
<td>63.3%</td>
</tr>
</tbody>
</table>
The Crowd is Un-wise
The Crowd is Un-wise

Temporal methods tried:
- Shortest path
- Min Time
- Expected Success
- Reinforcement learning
- Most Common Next
- Most Popular Path

18%
45%
12%
Desirable Path Algorithms

Poisson Common Path

\[ \gamma(s) = \text{first step in the *most frequent* path to the solution from } s, \text{ taken by *average* students. Assume poison process.} \]
Desirable Path Algorithms

**Poisson Common Path**

\[ \gamma(s) = \arg \min_{p \in Z(s)} \sum_{x \in p} \frac{1}{\lambda x} \]

- **Path Cost**
- **Submission count of partial solution**
- **Predicted next partial solution**
- **Paths to solution**
- **Partial solutions in the path**
## Evaluation

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$P_A$ Accuracy</th>
<th>$P_B$ Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>8.2%</td>
<td>4.3%</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>49.5%</td>
<td>33.6%</td>
</tr>
<tr>
<td>Min Time</td>
<td>67.2%</td>
<td>42.2%</td>
</tr>
<tr>
<td>Rivers Policy‡</td>
<td>72.9%</td>
<td>78.2%</td>
</tr>
<tr>
<td>Expected Success</td>
<td>77.9%</td>
<td>56.2%</td>
</tr>
<tr>
<td>MDP‡</td>
<td>80.5%</td>
<td>47.6%</td>
</tr>
<tr>
<td>Most Common Next</td>
<td>81.1%</td>
<td>49.0%</td>
</tr>
<tr>
<td>Static Analysis‡</td>
<td>86.3%</td>
<td>-</td>
</tr>
<tr>
<td>Most Popular Path</td>
<td>88.3%</td>
<td>52.8%</td>
</tr>
<tr>
<td>Ability Model</td>
<td>88.4%</td>
<td>63.3%</td>
</tr>
<tr>
<td>Independent Probable Path‡</td>
<td><strong>95.5%</strong></td>
<td>*83.3%</td>
</tr>
<tr>
<td>Poisson Path‡</td>
<td><strong>95.9%</strong></td>
<td>*84.6%</td>
</tr>
</tbody>
</table>
Story 4: Process has predictive power
State of the Art of this well studied problem

SOTA uses RNNs to vectorize programs and classify among K feedback classes.

These models are ...

- Far from human accuracy.
- Uninterpretable... why pick the feedback it did?
- Require lots of labeled examples.

Wang et al, EDM 2017
What Matters for Students?

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt

4. Completes attempt
5. Backtracks
6. Finds solution

Lisa Wang, Chris Piech. Deep Knowledge Tracing on Programming Exercises. L@S 2017
Highly Rates Grit

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt
4. Completes attempt
5. Backtracks
6. Finds solution

Lisa Wang, Chris Piech. Deep Knowledge Tracing on Programming Exercises. L@S 2017
Highly Rates Grit

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt

4. Completes attempt
5. Backtracks
6. Finds solution

Lisa Wang, Chris Piech. Deep Knowledge Tracing on Programming Exercises. L@S 2017
Highly Rates Grit

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt
4. Completes attempt
5. Backtracks
6. Finds solution

Lisa Wang, Chris Piech. Deep Knowledge Tracing on Programming Exercises. L@S 2017
Highly Rates Grit

1. Two compound errors
2. Solves first error
3. Starts reasonable attempt
4. Completes attempt
5. Backtracks
6. Finds solution

Lisa Wang, Chris Piech. Deep Knowledge Tracing on Programming Exercises. L@S 2017
Unachieved goal: understand a student based on their process

But we now have rubric sampling!
What would be the first step?